EventMe

Carina Erlebach, David Mischitz, Caitlin Rösen

June 2021

1 Introduction

The first idea for our application was given through partyholic. Our Main idea was to create an application to help one organising and not forgetting an event seen on e.g. an advertisement. Basically the application shall use a new picture taken or from the gallery and analyse any text on it. We are specifically looking for dates and a related location for the event. In our research we found several many calendar or event planning applications. None of them were for event management in that way we designed our to help manage and organize them e.g. connecting the app with one's Google calendar and instantly adding the event. The way our application sets events and finds the information is seemingly the only one at the moment. Thus it appears we found a gap in the market. After talking to one another and figuring out how we wanted the application to work we arranged us with what was to do.

2 Main

2.1 Design

A good app design can improve the overall user experience. Nice interfaces, noteworthy simplicity and easy navigation are fairly prominent traits of a good app. The design can be part of why people come back to the app. Of course only secondarily. We decided to use uizard /cite https://uizard.io/ as it provides the possibility of sketching your wireframes and integrate them into uizard. Furthermore one can easily switch between themes thus being able to quickly exchange different things such as buttons in the wireframe model or the actual design. For the logo from which our application shall be reconized we used Adobe Fresco Version 2.7.5. The logo represents the main functions of the app – which are taking a picture and inserting the information into one's calendar. We also tried InVision Studio which could be used for animation but overall the understanding of the software was harder and more complicated than uizard.The idea behind the logo was to show the two main aspects of EventMe which are using a picture and basically integrating it into ones calendar. Afterwards the first thing to do was to work out how the app should look. The first drafts

were easily sketched out with Adobe Fresco and were wireframe models. Uizard allowed us to switch from the wireframe to an actual aspired design draft. This draft was the most simple version of what we wanted the application to look like. Thus we rearranged and resized some buttons and icons in order to make the app look more even. Afterwards we added the text fields we wanted to have, e.g. settings or home. When these basics were finished the next step was to add colour into our app. The initial thought was a bright red which draws the attention to the app overall. After some research we decided to use a less vibrant colour which now became a colour gradient which is seemingly calm. Nonetheless the gradient design we decided on still has red in it aswell as orange and a very light peach colour. In symbolism red and orange are the colours of not only health and technolgy but even more of leisure time and entertainment. [3] Moreover these are enthusiasic and emotion-handling colours which are used to appeal to an user to interact with the application.[2] For each button we decided on simple, known and easily reconizable icons to have an even better user interface and experience hence these icons are usually present and commonly used.

2.2 Frontend

The front end development is very important for a good application since this part is what the user actually sees and interacts with. Putting this into less words: the front end is the construction of the user experience in cooperative work with both the design and the backend.

2.2.1 Software

We build our front end with Android Studios which uses Java as a language. We chose Android Studio because it has a clean interface and is very easy to use for beginners. Android Studio supports java and kotlin but we chose java because we worked with it before and we were not comfortable with learning a fully new languages. Looking back we should have chosen kotlin because it is a shorter way than java. This could have helped us with time and bugs because it is less code.

2.2.2 Structure

First of all we just did the main design. After the design was implemented we added the logic. We mainly used empty activities for our screens. Activities are the interface where the user can interact. We have seven activities that give our app structure. First we have the splashscreen. We used two animations. One from the top and one from the bottom. After that we have the permissions screen which is just a backgroud and then the main screen. In the main screen we have two buttons which use intents. Intents are a google java class that helped us in many ways with the device functions. Here we used it to open the camera, gallery and to save the pictures into a bitmap. After taking or selecting

a picture you will get to the confirmation screen. Here you can see the bitmap which holds the chosen image and have two buttons as well. You can either go back to the main screen with a contentView method or it will send the picture to the server with a https request. You will also see the loading screen. After the loading screen you will get to the results screen. You can either go back to the main screen or you can add your event to the calendar. We use a intent again to excess the google calendar. That is the basic logic behind our app.

2.2.3 Problems

A mayor problem we faced were fragments. In the early stages of the app we wanted to do a drawer layout. Drawer layouts use fragments to create the navigation. First it started with buttons that did not work correctly in the fragment. With a normal activity you can just do a global java class that every button onclick can use. Sadly fragments do not work like that because it is just the UI of an activity. So we switched our whole design to a activity based app which got rid of the button problem. A big problem was taking a picture. Due to not having an android we used a emulator to test the app and it did too much on the main thread and skipped to many frames to work properly. We decided to go line with line and gave out debugging massages it worked finally. We used a lot of Youtube videos and the project from last year to overcome our problems.

2.3 Backend

While the front end is what the user sees the backend is rather the, backround player‘ and manages how the show is run. Mainly the backend contains the server which provides the data upon requests sent, the database organizing the information and the application channeling it.

2.3.1 Frameworks and API

We used Flask as a way to create a simple web application, as it does not require particular tools or libraries. We deployed that application to Heroku which is a cloud platform using containers to run our application on a virtual environment. We tested different iterations of the application using separate containers, pushing our code directly from GitHub. The most important task of our backend is the image processing for this we used Google Cloud Vision. It is a powerful and reliable API that analyses images via deep learning. Vast quantities of training data provided from Google makes it better than most similar APIs

2.3.2 Structure

The frontend sends an binary decoded image to the backend server by writing it into an output stream . Once the file stream is red the image is temporarily saved to the server. The image is then sent to the Google Cloud server where

it is processed. According to the function the text on the image is sent back in JSON format. To extract the relevant information from the text regular expressions are used. As response to the frontend the data is sent in a .json file.

2.3.3 Problems

On the backend side our biggest Problems came with the API used for optical character recognition. We started out using tesseract because it was completely free and very easy to use. The first issue using tesseract was deploying it to heroku we had difficulties locating the important files once we uploaded them to the the server using a buildpack. Simply downgrading the operating system of our server solved the issue. A much bigger issue was that tesseract was not providing good enough results, it only worked with easy to read black text on a white background. Most Posters do not fulfill does requirements. So we needed a more versatile and reliable API. This API was Cloud Vision. It provides vastly improved results over tesseract and has many more possibilities for image processing. But Cloud Vision also comes with a downside. The Service is not completely free. At a certain number of requests every additional one will cost a bit of money. For this reason we decided not putting the app on the Google Play Store.

3 Conclusion

Our future ideas for EventMe include - in addition to the Google Calendar - Google Maps. This integration would give the possibility of having an easy guidance to the event location. Not only would the user be reminded by their calendar that an event is occuring but even more they would know where it is happening and how to get their. Moreover at the current point EventMe is solely accessible for Android systems thus to broaden the possible usage we'd like to make the application usable for iOS systems. For the overall usability and user experience a more reliable extraction of event title and location is inevitable. To achieve this we need to take a look into more advanced NLP methods and try to include these. The application can be downloaded from our github. [1] We used github as it is an open source repository service which allows to store code in a cloud like fashion. Not only could the code be uploaded by any collaborator but even more can be checked by a variety of people who then even can tell us what we've overlooked or forgotten or even errors can be pointed out by the github community. This way we can learn from professionals or people with more coding experience and exchange first drafts easily and with a certain feedback.

4 GitHub

https://github.com/caiidar/Event-Planer-NLP

References

[1] Caitlin Rösen Carina Erlebach, David Mischitz. Event-planer-nlp.

[2] none mentioned. App design: Die bedeutung und wirkung der farben.

[3] James Thomas. Farben kombinieren – so verbesserst du die usability deiner webseite.